



Lecture 20: While Loops (Sections 7.3, 7.4)

CS 1110

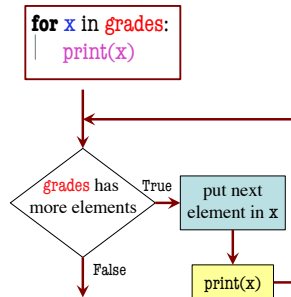
Introduction to Computing Using Python



Cornell CIS
COMPUTING AND INFORMATION SCIENCE

[E. Andersen, A. Bracy, D. Gries, L. Lee, S. Marschner, C. Van Loan, W. White]

Recall: For Loops



- **loop sequence:** `grades`
- **loop variable:** `x`
- **body:** `print(x)`

To execute the for-loop:

1. Check if there is a “next” element of **loop sequence**
2. If so:
 - *assign* next sequence element to **loop variable**
 - Execute all of **the body**
 - Go back to Line 1
3. If not, terminate execution₃

Different types of Repetition

1. Process each item in a sequence

- Compute statistics for a dataset.
- Send all your contacts an email.

```

for x in sequence:
    process x
    
```

2. Do something *n* times

- Draw a checkers board.
- Run a protein-folding simulation for 10⁶ time steps.

```

for x in list(range(n)):
    do something
    
```

3. Do something an unknown number of times

- Fly up until you're near the ceiling.
- Play hangman until 6 strikes.

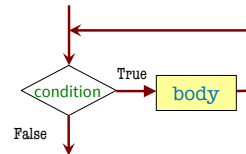


Beyond Sequences: The while-loop

```

while <condition>:
    statement 1
    ...
    statement n
    
```

} body



- Relationship to for-loop
 - Broader notion of “keep working until done”
 - Must explicitly ensure condition becomes false
 - *You* explicitly manage what changes per iteration

While-Loops and Flow

```

import random
num = random.randint(0,10)
guessed_it = False
print('I'm thinking of a number.')
guessed_it = False
while not guessed_it:
    guess = input('Guess it: ')
    guessed_it = (num == guess)
print('Well done!')
    
```

```

I'm thinking of a number.
Guess it: 6
Guess it: 2
Guess it: 1
Guess it: 4
Well done!
    
```

Q1: What gets printed?

<pre> a = 0 while a < 1: a = a + 1 print(a) </pre>	<pre> a = 0 while a < 2: a = a + 1 print(a) </pre>	<pre> a = 0 while a > 2: a = a + 1 print(a) </pre>
□	□	□

Q2: What gets printed?

```
a = 4
while a > 0:
    a = a - 1
```

```
print(a)
```



```
a = 0
while a < 3:
    if a < 2:
        a = a + 1
```

```
print(a)
```



9

Q3: What gets printed?

```
a = 8
b = 12
while a != b:
    if a > b:
        a = a - b
    else:
        b = b - a
print(a)
```

- A: Infinite Loop!
- B: 8
- C: 12
- D: 4
- D: I don't know

11

for vs. while

do something n times

```
for k in list(range(n))
    # do something
```

```
k = 1
while k < n:
    # do something
    k = k+1
```

Must remember to increment

My preference? for-loop

14

for vs. while

do something an unknown number of times

```
for k in list(range(BIG_NUM))
    # do something
if time to stop:
    break
```

```
while not time to stop:
    # do something
```

My preference? while-loop

15

Remember Hangman?

```
import random, hangman
word_list = [ ... words we want user to guess .. ]
N_GUESSES = 10
secret = hangman.SecretWord(random.choice(word_list))

for n in list(range(N_GUESSES)):
    secret.word_so_far()
    user_guess = input("Guess a letter: ")
    secret.apply_guess(user_guess):
    if secret.is_solved():
        print("YOU WIN!!!")
        break #jumps out of the for-loop
secret.reveal()
```

Usually can keep guessing until you get some number wrong

20

Using while-loops Instead of for-loops

Advantages

- Better for **modifying data**
 - More natural than range
 - Works better with deletion
- Better for **convergent tasks**
 - Loop until calculation done
 - Exact steps are unknown
- Easier to **stop early**
 - Just set loop var to False

Disadvantages

- **Infinite loops** more likely
 - Easy to forget loop vars
 - Or get stop condition wrong
- **Require** more management
 - Initialize the condition?
 - Update the condition?

22